Meta Learning with Implicit Gradients

Aravind Rajeswaran*[1], Chelsea Finn*[2], Sham Kakade[1], Sergey Levine[3]

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING · Stanford University · BAIR BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Introduction

The standard machine learning paradigm

Successful if **easy**, **cheap**, and **safe** to collect **large** amount of data.

For many applications like robotics, user personalization, or low resource translation, big-data is non-existent, costly, or sensitive.

Meta Learning: Learning algorithmic procedures which enable efficient learning of new tasks by encoding adaptable representations

Problem Setting

i	*task index (# tasks = N)*	ϕ	*model (weights)*
θ	*meta parameters (init, lr, #steps)*	$\mathscr{A}$	*learning algorithm (e.g. SGD)*
$\mathscr{L}$	*loss function*	$\mathscr{D}$	*task (training) dataset*

$$\min_{\theta} \left\{ F(\theta) := \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}_i \left(\phi_i = \mathscr{A}(\mathscr{D}_i, \theta) \right) \right\}$$

Learn a set of meta parameters θ^* that make $\mathscr{A}$ behave efficiently

$$\nabla_\theta \mathscr{L}_i(\theta) = \frac{d\phi_i}{d\theta} \nabla_\phi \mathscr{L}_i(\phi_i)$$

For new task$(\tau): \phi_\tau = \mathscr{A}(\mathscr{D}_\tau, \theta^*)$ *hard to compute*, *easy to compute*

Idea: Optimize $F(\theta)$ through gradient based iterative algorithms.

Requirement: Efficient computation of task meta-gradients $\nabla_\theta \mathscr{L}_i(\theta)$

MAML [1]: Backpropagate through the iterative steps of $\mathscr{A}$

- ***Restricts algorithms:*** each atomic operation needs to be first order and differentiable (no line-search, trust-region, randomization)
- ***Memory complexity*** is linear in the length of $\mathscr{A}$ (can't optimize well)
- ***Vanishing gradients*** when backpropagating through long paths

The Implicit MAML Algorithm

Interpret as bi-level optimization and use implicit function theorem

Short hands : $\mathscr{L}_i(\phi) \equiv \mathscr{L}_i(\phi, \mathscr{D}_i^{test})$ and $\hat{\mathscr{L}}_i(\phi) \equiv \mathscr{L}_i(\phi, \mathscr{D}_i^{tr})$

$$\min_{\theta} \ F(\theta) := \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}_i \left(\mathscr{A}_i^*(\theta) \right) \quad \text{where}$$

$$\mathscr{A}_i^*(\theta) := \arg\min_{\phi} G_i(\phi, \theta) = \hat{\mathscr{L}}_i(\phi) + \frac{\lambda}{2} \|\phi - \theta\|_2^2$$

Regularization leads to non-vanishing gradient and analytical expression

***Lemma** (Implicit gradient): Let $\phi_i^* := \mathscr{A}_i^*(\theta)$. Then, we have*

$$\nabla_\theta \mathscr{L}_i(\theta) = \left(I + \frac{1}{\lambda} \nabla_\phi^2 \hat{\mathscr{L}}_i(\phi_i^*) \right)^{-1} \nabla_\phi \mathscr{L}_i(\phi_i^*)$$

Note : The gradient depends only on the result of $\mathscr{A}$ and not the path!

Practical Algorithm (Implicit MAML or iMAML)

1. Solve inner optimization approximately to find $\|\phi_i - \phi_i^*\| \leq \delta$
2. Approximately find meta-gradient using conjugate gradient algorithm that requires only Hessian-vector products to get $\left\|g_i - \left(I + (1/\lambda) \nabla_\phi^2 \hat{\mathscr{L}}_i(\phi_i) \right)^{-1} \nabla_\phi \mathscr{L}_i(\phi_i) \right\| \leq \delta$

***Theorem** (error is controllable) If $G_i(\phi, \theta)$ is strongly convex in ϕ, for above algorithm, we have bounded error $\|g_i - \nabla_\theta \mathscr{L}_i(\theta)\| \leq O(\delta)$*

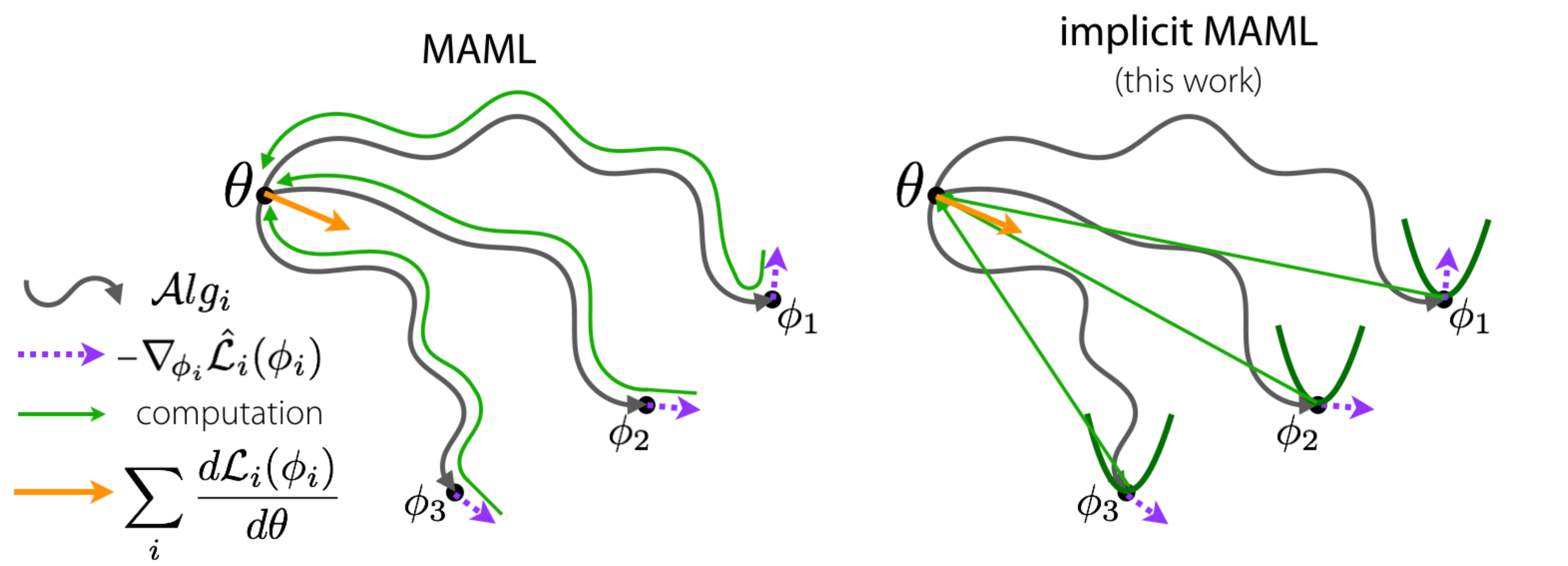

Table: *Compute and memory complexity. D = diameter, κ = condition number of inner level. † compares with $\mathscr{A}$, while * compares with $\mathscr{A}^*$*

Algorithm	Compute	Memory	Error
MAML (GD + full back-prop)	$\kappa \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla \hat{\mathcal{L}}_i) \cdot \kappa \ \log\left(\frac{D}{\delta}\right)$	$0^\dagger$
MAML (Nesterov's AGD + full back-prop)	$\sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla \hat{\mathcal{L}}_i) \cdot \sqrt{\kappa} \ \log\left(\frac{D}{\delta}\right)$	$0^\dagger$
Truncated back-prop (GD) [2]	$\kappa \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla \hat{\mathcal{L}}_i) \cdot \kappa \ \log\left(\frac{1}{\epsilon}\right)$	$\epsilon^\dagger$
Implicit MAML (this work)	$\sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla \hat{\mathcal{L}}_i)$	δ^*

Experiments

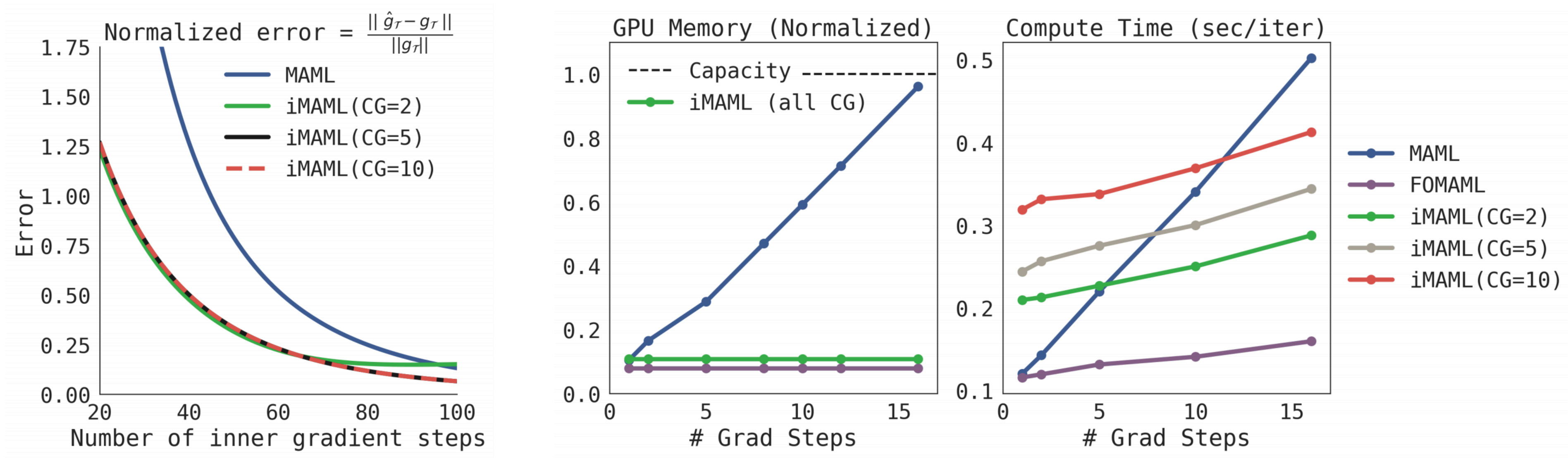

Figure: *(left) MAML and iMAML computation vs exact meta-gradient on a synthetic example. (right) Compute and memory on 20-way-5-shot Omniglot*

Table: *Comparison of algorithms on Omniglot. Gradient descent (GD) and Hessian-Free (w/ line-search) algorithms considered for $\mathscr{A}$. $\lambda = 2.0$ and CG=5*

Algorithm	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
MAML [15]	$98.7 \pm 0.4\%$	$\mathbf{99.9 \pm 0.1\%}$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$
first-order MAML [15]	$98.3 \pm 0.5\%$	$99.2 \pm 0.2\%$	$89.4 \pm 0.5\%$	$97.9 \pm 0.1\%$
Reptile [43]	$97.68 \pm 0.04\%$	$99.48 \pm 0.06\%$	$89.43 \pm 0.14\%$	$97.12 \pm 0.32\%$
iMAML, GD (ours)	$99.16 \pm 0.35\%$	$99.67 \pm 0.12\%$	$94.46 \pm 0.42\%$	$98.69 \pm 0.1\%$
iMAML, Hessian-Free (ours)	$\mathbf{99.50 \pm 0.26\%}$	$99.74 \pm 0.11\%$	$\mathbf{96.18 \pm 0.36\%}$	$\mathbf{99.14 \pm 0.1\%}$

Summary

- ***No vanishing meta-gradients*** due to use of regularization
- Meta-gradient depends only on ***final result of algorithm, not path***
- ***Wider class of algorithms*** are supported by implicit MAML
- Implicit MAML is provably ***efficient*** in computation and memory, provably ***convergent***, and leads to ***empirical gains*** on benchmarks
- FOMAML and Reptile are CG=0 approximations of iMAML

(* equal contributions, [1]University of Washington, [2]Stanford, [3]UC Berkeley)

[1] Finn, Abbeel, Levine. ICML 2017. [2] Shaban et al. AISTATS 2019.